Министерство просвещения Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Набережночелнинский государственный педагогический университет»

Факультет математики и информатики

Кафедра информатики и вычислительной математики

КУРСОВОЙ ПРОЕКТ

на тему:

**РАЗРАБОТКА И РЕАЛИЗАЦИЯ МОДУЛЯ УПРАВЛЕНИЯ РАБОТОЙ РЕГИСТРАТУРЫ ПОЛИКЛИНИКИ**

Направление подготовки 09.03.03 Прикладная информатика в дизайне

|  |  |  |
| --- | --- | --- |
| Руководитель  Кандидат экономических наук, доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | Хамидуллин М.Р. |
|  |  |  |
| Обучающийся | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | Шафигуллина Э.Р.  Номер группы 2021 |

Набережные Челны, 2024

Оглавление

[Введение 3](#_Toc177356519)

[1. Анализ текущего состояния регистратуры поликлиники 4](#_Toc177356520)

[1.1 Анализ процесса приема и обслуживания 4](#_Toc177356521)

[1.2 Анализ требований к продукту 5](#_Toc177356522)

[1.2.1 Проблемы современного программного обеспечения 5](#_Toc177356523)

[1.2.2 Функциональные требования 6](#_Toc177356524)

[1.2.3 Нефункциональные требования 7](#_Toc177356525)

[1.2.4 Требуемое программное обеспечение 8](#_Toc177356526)

[1.2.5 Требуемое аппаратное обеспечение 9](#_Toc177356527)

[1.2.6 Требуемый набор технологий программного обеспечения 10](#_Toc177356528)

[2. Раздел. Разработка программного обеспечения 13](#_Toc177356529)

[2.1 Разработка модели проектируемого ПО 13](#_Toc177356530)

[2.2 Функциональная модель 15](#_Toc177356531)

[2.3 Прецедентная модель 17](#_Toc177356532)

[2.4 Объектно-ориентированная модель 19](#_Toc177356533)

[Заключение 20](#_Toc177356534)

[Список использованной литературы 21](#_Toc177356535)

[Приложения 22](#_Toc177356536)

[Приложение А – Код создания базы данных 22](#_Toc177356537)

# Введение

Современная система здравоохранения сталкивается с многочисленными вызовами, среди которых важное место занимает организация эффективной работы регистратуры поликлиники. Регистратура является первым звеном взаимодействия пациента с медицинским учреждением, и от её работы во многом зависит удовлетворенность пациентов и общая эффективность медицинской помощи.

Актуальность данной темы обусловлена тем, что в условиях растущего числа пациентов и увеличения объема информации, которая должна быть обработана, традиционные методы управления регистратурой уже не справляются с поставленными задачами. Внедрение информационных технологий и разработка специализированного модуля управления работой регистратуры позволяет существенно улучшить качество обслуживания, сократить время ожидания, оптимизировать работу медицинского персонала и повысить общую эффективность медицинского учреждения.

Целью данного курсового проекта является разработка и реализация модуля управления работой регистратуры поликлиники, который будет включать в себя функции записи на прием, ведения электронной медицинской карты пациента, автоматизации процесса обработки запросов и обращений, а также предоставления аналитических данных для улучшения качества обслуживания.

В ходе работы будут рассмотрены современные подходы к автоматизации работы регистратуры, проведен анализ существующих решений, выявлены их недостатки и предложены пути их преодоления. Основное внимание будет уделено разработке практических рекомендаций по внедрению и использованию разработанного модуля в условиях реальной поликлиники.

# Анализ текущего состояния регистратуры поликлиники

# Анализ процесса приема и обслуживания

В настоящее время процесс приема и обслуживания пациентов в регистратуре поликлиники включает в себя несколько этапов:

* Запись на прием ­— пациенты могут записаться на прием по телефону, лично или через интернет-портал. Однако существующие системы часто не синхронизированы между собой, что приводит к дублированию информации и ошибкам в расписании;
* Регистрация — после прибытия в регистратуру пациента регистрируют, оформляя медицинскую карту, если она отсутствует, или обновляя существующую. Оформление медицинской карты может занять значительное время, особенно если пациент обращается впервые;
* Выдача направлений — при необходимости пациенту выдают направление на обследование или консультацию к другому специалисту. Процесс оформления направлений может быть затруднен из-за отсутствия единой базы данных о доступных врачах и обследованиях;
* Оформление медицинской документации — в регистратуре пациенту предоставляют информацию о необходимом оформлении документации и процедурах. Процесс оформления часто затягивается из-за бумажной волокиты и отсутствия автоматизированных систем;
* В результате множества ручных операций и обработки бумажной документации, время, затрачиваемое на регистрацию пациента, может значительно варьироваться от нескольких минут до получаса, в зависимости от сложности запроса и загруженности персонала. Это приводит к регулярным задержкам и осложняет координацию работы.

# Анализ требований к продукту

# Проблемы современного программного обеспечения

В большинстве поликлиник используются устаревшие информационные системы или бумажные носители для ведения расписания, учета пациентов и предоставления информации персоналу.

Наличие различных систем, не связанных между собой, не только затрудняет обмен данными, но и повышает риск ошибок и потерь информации. Отсутствие единой интегрированной системы приводит к неэффективности работы, усложняет ведение отчетности и затрудняет анализ данных. Можно выделить следующий перечень проблем:

* Системы записи на прием могут быть несовместимы друг с другом, что затрудняет обмен данными и координацию работы между различными отделами поликлиники.
* Системы обработки платежей, если они имеются, могут не быть удобными для пациентов и персонала, что приводит к задержкам в обслуживании.
* Электронные медицинские карты, если они используются, часто не интегрированы с другими системами, что приводит к дублированию информации и затрудняет доступ к данным для различных специалистов.
* Программное обеспечение поликлиники включает в себя комплекс программных продуктов, которые используются для управления различными процессами.

# Функциональные требования

В следствии анализа аналогов получилось выделить набор типов программного обеспечения:

* Системы записи на прием. Важными функциями таких систем являются:

1. Учет свободных слотов у врачей;
2. Управление расписанием приема;
3. Интеграция с системами записи на обследования и анализы;
4. Возможность получения информации о пациенте.

* Электронные медицинские карты (ЭМК). ЭМК — это цифровые носители медицинской информации о пациенте, которые хранят всю информацию о его здоровье:

1. Анамнез пациента;
2. Результаты обследований;
3. Назначения лекарств;
4. История болезни.

* Системы обработки платежей. Эти системы используются для приема платежей от пациентов за медицинские услуги:

1. Оплата за прием;
2. Оплата за обследования;
3. Оплата за лекарственные препараты.

* Системы управления ресурсами, такими как:

1. Кадры;
2. Оборудование;
3. Финансы.

* Системы управления документами, для таких документов, как:

1. Справки;
2. Больничные листы;
3. Медицинские заключения.

# Нефункциональные требования

Стоит выделить и нефункциональные требования к программному обеспечению:

* Безопасность и конфиденциальность данных: программное обеспечение должно обеспечивать высокий уровень безопасности и конфиденциальности данных о пациентах;
* Интеграция с другими системами: программное обеспечение должно быть интегрировано с другими системами поликлиники, такими как системы записи на прием, ЭМК, системы управления ресурсами;
* Простота использования: программное обеспечение должно быть интуитивно понятным и простым в использовании как для персонала, так и для пациентов;
* Мобильность: возможность доступа к данным и функционалу системы с мобильных устройств;
* Производительность: быстрый ответ системы на действия пользователя;
* Отказоустойчивость: возможность системы выдерживать достаточное количество одновременных запросов от пользователей;
* Масштабируемость: возможность легко расширять систему горизонтально.

# Требуемое программное обеспечение

В результате анализа для реализации может быть использовано следующее дополнительное программное обеспечение:

* Система управления ресурсами поликлиники (ERP):

1. 1C: Здравоохранение — система для учета медицинских и финансовых операций, управления персоналом и медицинским оборудованием;
2. NextGen Healthcate — популярная система с множеством решений популярная в Европе.

* Система управления отношениями с пациентами (CRM):

1. Bpm'online healthcare — платформа для управления взаимодействием с пациентами, маркетингом и обслуживанием;
2. Salesforce Health Cloud — версия популярной CRM системы Salesforce, специализированная для сферы здравоохранения.

* Система управления документооборотом:

1. Docsvision: автоматизация работы с документами;
2. 1С: Документооборот — популярная на российском рынке система, с легкостью интегрируемая с другими продуктами 1С;
3. Directum — комплексная система управления документами, также позволяет управлять бизнес-процессами.

# Требуемое аппаратное обеспечение

Исходя из выше указанных требований к ПО, требуется следующее аппаратное обеспечение:

* Серверное оборудование:

1. HPE ProLiant DL Servers: надежные серверы для хранения больших объемов медицинских данных и обеспечения непрерывной работы МИС;
2. Cisco UCS: универсальная система, обеспечивающая высокую производительность.

* Рабочие станции и персональные компьютеры:

1. Dell OptiPlex: для административного персонала и врачей для доступа к МИС и ведения электронной документации;
2. HP ProDesk: может подойти как аналог предыдущего варианта, обеспечивающий хорошее сочетание производительности и надежности.

* Мобильные устройства: планшеты и смартфоны для возможности доступа к системе в любой точке поликлиники;
* Сетевое оборудование Cisco Networking Equipment для обеспечения стабильной и безопасной работы внутренней сети учреждения.

# Требуемый набор технологий программного обеспечения

При выборе языка программирования и системы управления базами данных (СУБД) для разработки медицинских приложений важно учитывать их возможности для обеспечения безопасности, производительности и масштабируемости.

Язык программирования Python может быть привлекательным для разработки подобных систем благодаря своей простоте в использовании и широкому набору библиотек для обработки данных и выполнения аналитики. Однако, язык C# и среда .NET предлагают ряд преимуществ, которые делают их более подходящими для создания крупных и критически важных приложений. C# предоставляет более строгую типизацию и широкие возможности обработки массивов данных, что способствует созданию надежных и масштабируемых решений. Среда .NET обеспечивает высокую производительность и мощные средства для построения безопасных приложений, что особенно важно для работы с медицинскими данными, где ошибки могут иметь серьезные последствия.

В отношении СУБД, Microsoft SQL Server демонстрирует высокую производительность и надежность при работе с большими объемами данных, что важно для медицинских приложений, обрабатывающих огромные массивы информации. Он также предлагает продвинутые функции для обеспечения безопасности данных. В то время как PostgreSQL и MySQL имеют свои достоинства, SQL Server обеспечивает лучшую интеграцию с .NET и комплексные возможности для поддержки транзакций, что делает его более подходящим для сложных и высоконагруженных систем.

Таким образом, выбор C# и .NET в сочетании с Microsoft SQL Server наиболее оправдан для создания медицинских приложений, благодаря их способности обеспечить высокую производительность, безопасность и масштабируемость, соответствующие требованиям современного медицинского учреждения.

**1.3 Предложения по улучшению деятельности поликлиники**

В ходе анализа также удалось выявить предложения по улучшению деятельности клиники.

Предлагаемые улучшения по автоматизации записи на прием:

* Удобный онлайн-сервис: создать интуитивно понятный веб-сайт или мобильное приложение, которое позволит пациентам записываться на прием к врачу онлайн, выбирая удобное время и врача;
* Интеграция с ЭМК: обеспечить синхронизацию модуля записи с ЭМК, чтобы автоматически вводить информацию о пациенте и его истории болезни;
* Система "умного" расписания: разработать алгоритм, который учитывает загруженность врачей, специфику их приема (например, консультации или процедуры) и позволяет оптимизировать распределение пациентов;
* Возможность отмены записи: создать функцию отмены записи, чтобы пациенты могли изменять свои планы без необходимости обращаться в регистратуру лично.

Предлагаемые улучшения управления электронными медицинскими картами (ЭМК):

* Единая база данных: создать единую базу данных для ЭМК, доступную для всех специалистов, которые обрабатывают информацию о пациенте;
* Автоматизация ввода данных: разработать инструменты для упрощения ввода информации в ЭМК, например, шаблоны для стандартных записей, автоматическое заполнение полей;
* Защита данных: обеспечить шифрование и авторизацию доступа к ЭМК, чтобы защитить конфиденциальность информации о пациентах;
* Мобильный доступ: разрешить медицинскому персоналу доступ к ЭМК с мобильных устройств.

Предлагаемые улучшения автоматизации обработки запросов:

* Система обработки запросов: создать систему для автоматической обработки запросов от пациентов, например, на выдачу направлений, запросов на информацию о медицинских услугах или запросов на оформление документов;
* Электронные направления: внедрить систему электронных направлений, которая автоматически генерирует направление и передает его в соответствующий отдел;
* Шаблоны документов: создать шаблоны для стандартных медицинских документов (например, справки, больничные листы), чтобы автоматизировать процесс их оформления;

Предлагаемые улучшения управление финансовыми операциями:

* Электронная оплата: Внедрить систему онлайн-оплаты для пациентов, чтобы они могли оплатить медицинские услуги без посещения кассы;
* Автоматизация выписки счетов: Создать систему автоматической выписки счетов за медицинские услуги.

Предлагаемые улучшения статистики и аналитики:

* Система сбора статистики: Разработать систему сбора статистики о работе регистратуры, например, о количестве пациентов, времени ожидания, распределении пациентов по врачам;
* Генерация отчетов: Создать возможность генерации отчетов о работе регистратуры в разных форматах;
* Анализ данных: Встроить функции для анализа данных о работе регистратуры с целью идентификации узких мест и оптимизации процессов.

# Раздел. Разработка программного обеспечения

# Разработка модели проектируемого ПО

Проектируемое программное обеспечение для регистратуры клиники предназначено для автоматизации процессов учета и управления записями пациентов, взаимодействия с медицинским персоналом и обработки медицинской информации. Основная цель разработки — создание интуитивно понятной и эффективной системы, которая упростит процессы записи на прием, улучшит взаимодействие между пациентами и медицинскими работниками, а также повысит качество и доступность медицинских услуг.

Основные функции проектируемого ПО:

* Автоматическое и ручное создание записей на прием;
* Оптимизация расписания врачей с учетом их специализации и рабочего времени.
* Управление данными о врачах (специализация, график работы, квалификация);
* Мониторинг загруженности врачей и распределение рабочего времени.
* Централизованное хранение медицинских карт пациентов;
* Обеспечение конфиденциальности и безопасности данных.
* Обмен данными с лабораторными информационными системами (LIS) и системами электронного здравоохранения (EHR).
* Формирование отчетов о работе клиники, загруженности врачей и статистике посещений;
* Анализ данных для улучшения качества обслуживания и планирования работы клиники.

Для создания базы данных необходимо разработать правильную модель, удовлетворяющую функциональным требованиям. После анализа указанных выше требований, разработана качественная модель данных.

Таблицы базы данных:

* Patients — таблица для хранения информации о пациентах (ID пациента, ФИО, контактные данные, медицинская карта и т.д.);
* Doctors — таблица для учета врачей (ID врача, специализация, график работы и т.д.);
* Appointments — таблица для учета записей на прием (ID записи, ID пациента, ID врача, дата и время приема и т.д.).

Связи между таблицами:

* Связь между таблицами Patients и Appointments (один пациент может иметь несколько записей на прием);
* Связь между таблицами Doctors и Appointments (один врач может проводить несколько приемов).

Также следует предусмотреть сценарий работы системы:

* Создание записи на прием: Пациент выбирает врача и время приема через интерфейс приложения.
* Подтверждение записи: Система автоматически подтверждает запись и отправляет уведомление пациенту.
* Прием у врача: Врач проводит прием, используя данные из медицинской карты пациента.
* Отчетность: По завершении рабочего дня система формирует отчеты для администрации клиники.

Реализация данного программного обеспечения позволит улучшить управление регистратурой клиники, сделать процесс записи на прием более удобным и доступным для пациентов, а также повысить эффективность работы медицинского персонала.

# Функциональная модель

Диаграмма IDEF0 (Integration Definition for Function Modeling) предназначена для визуального представления функций и процессов в системе. Она помогает понять, как функции взаимодействуют, какие входные данные и выходы они имеют, и какие ресурсы для их выполнения необходимы. Использование диаграммы IDEF0 упрощает анализ и оптимизацию процессов, служит для документации и упрощает управление изменениями, а также улучшает коммуникацию между участниками проекта. Это полезный инструмент для моделирования сложных систем и обеспечения ясности в их функционировании.

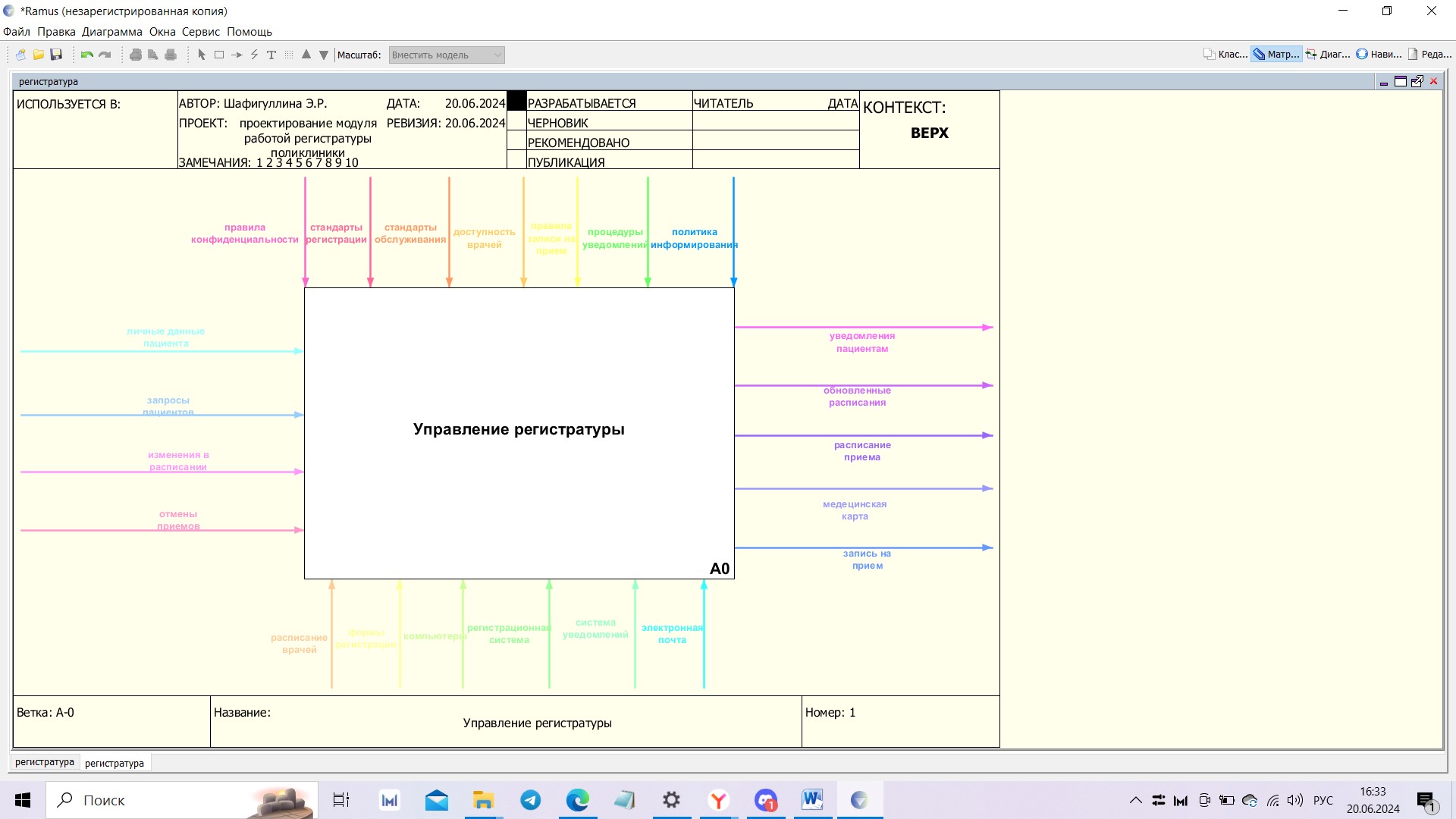


Рисунок 2.2.1 диаграмма IDEF0 для управления работой регистратуры поликлиники: уровень 1.

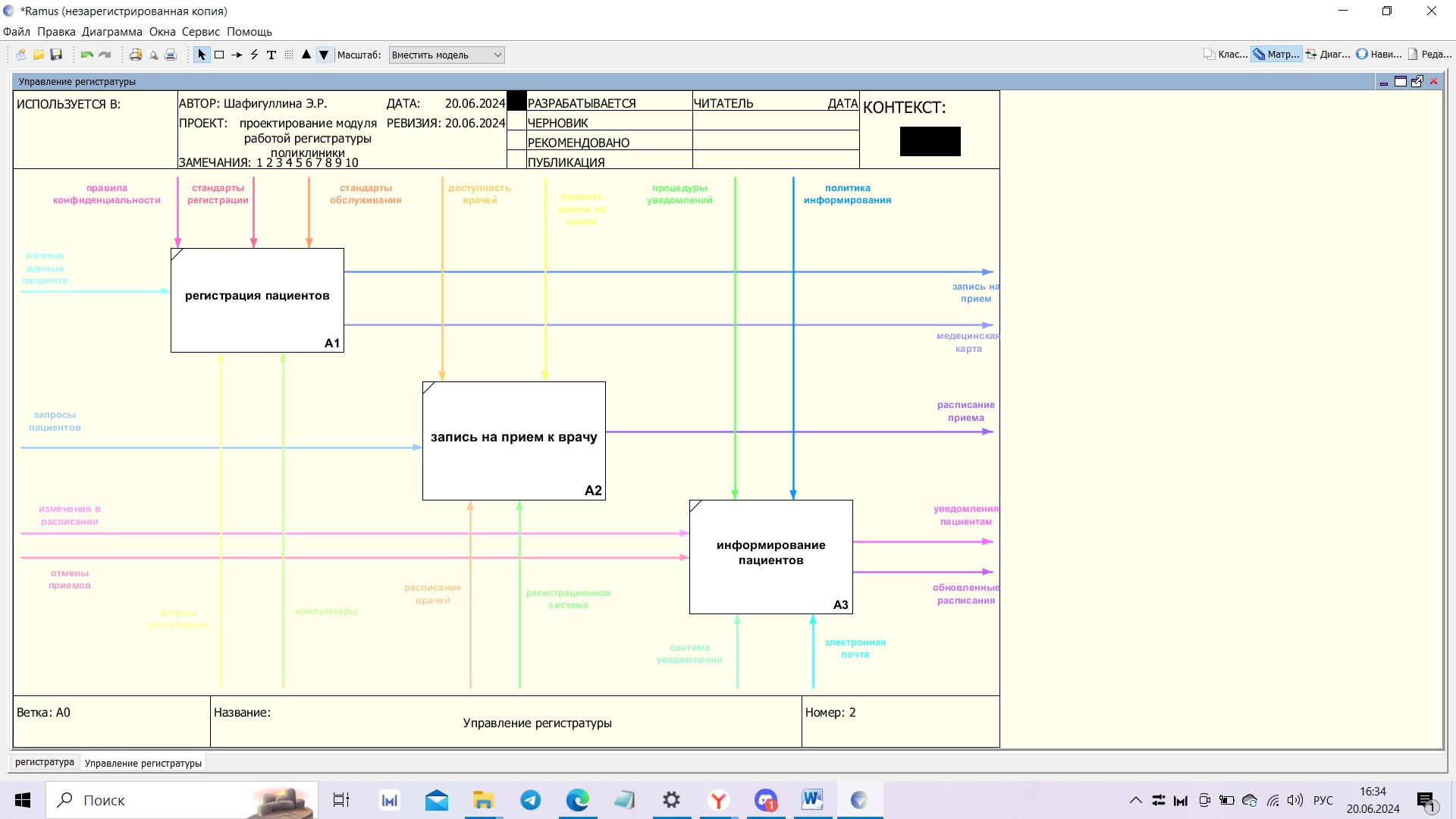
Рисунок 2.2.2 диаграмма IDEF0 для управления работой регистратуры поликлиники: уровень 2.

Диаграмма IDEF0 использования выглядит следующим образом:

* Уровень 0: Общий процесс управления регистратурой

1. A0 "Управление регистратурой"

* Уровень 1: Декомпозиция основных функций

1. A1 "Регистрация пациентов"
2. A2 "Запись на прием к врачу"
3. A3 "Информирование пациентов"

# Прецедентная модель

Диаграмма вариантов использования в UML — диаграмма, отражающая отношения между пользователями (акторами) и прецедентами. Она является составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Диаграмма вариантов использования в UML (Unified Modeling Language) используется для визуализации функциональных требований к системе. Она показывает, какие действия (варианты использования) могут выполнять пользователи (акторы) в рамках системы. Вот основные цели использования такой диаграммы:

Определение функциональности: Диаграмма помогает определить, что именно должна делать система с точки зрения пользователя.

Понимание пользовательских потребностей: Позволяет разработчикам увидеть систему глазами пользователя и лучше понять их потребности.

Коммуникация с заинтересованными сторонами: Упрощает общение между разработчиками, заказчиками и пользователями, предоставляя общий язык для обсуждения функционала

Планирование проекта: Служит основой для планирования разработки, тестирования и написания документации.

Анализ и моделирование: Помогает анализировать требования и моделировать поведение системы до её реализации.

Диаграмма вариантов использования является ключевым элементом в процессе сбора и анализа требований, а также в проектировании системы на ранних этапах разработки. Она обеспечивает ясность и структурированность, что важно для успешного создания и внедрения информационных систем.

Диаграмма прецедентов — это графическое представление взаимодействия между системой и ее окружением, а также между различными частями системы. Она позволяет понять, какие действия могут быть выполнены пользователями системы, какие функции системы могут быть использованы и как они взаимодействуют друг с другом.

Диаграмма прецедентов полезна для описания функциональности модуля информационной системы с точки зрения его пользователей-врачей и женщин. Она помогает идентифицировать основные действия, которые выполнены пользователями системы, определить их взаимосвязь и влияние на систему, а также выделяет основные функции модуля. На рисунке 1 представлена диаграмма прецедентов.

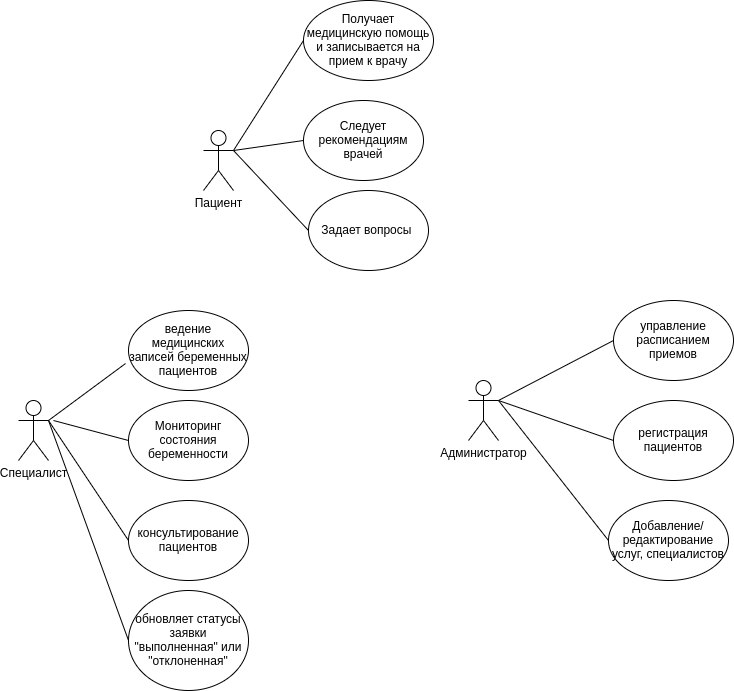


Рисунок 2.3.1 Диаграмма прецедентов

Такая диаграмма поможет лучше понять структуру регистратуры клиники и взаимодействие между ее компонентами.

# Объектно-ориентированная модель

Проектирование диаграммы классов ­— это процесс определения классов, их атрибутов и методов, а также связей между классами в рамках модуля информационной системы. Это позволяет создать структурированное представление компонентов модуля и их взаимодействия, что облегчает понимание его функциональности.

Диаграмма классов для данной курсовой работы будет полезна для описания структуры модуля информационной системы с точки зрения его компонентов и их взаимосвязей. Она поможет идентифицировать основные классы, их атрибуты и методы, определить связи между классами, а также выделить основные компоненты модуля. Диаграмма классов также поможет определить требования к системе, описать структуру модуля информационной системы и обеспечить понимание взаимодействия между его компонентами. Это важный инструмент для анализа и проектирования модуля информационной системы, который поможет создать понятную и наглядную модель его структуры и взаимодействия с окружением.

Изображение выглядит как текст, снимок экрана, диаграмма, число

Автоматически созданное описание

Рисунок 2.4.1 – Диаграмма классов

# Заключение

В рамках данного курсового проекта была осуществлена разработка и реализация модуля управления работой регистратуры поликлиники. Внедрение такого модуля позволяет автоматизировать и усовершенствовать процессы записи на прием к врачам, что способствует повышению качества медицинских услуг и удовлетворенности пациентов.

Проведен анализ текущего состояния регистратуры, выявлены ключевые проблемы и задачи, с которыми сталкиваются как пациенты, так и медицинский персонал. Были изучены существующие информационные системы и предложены пути их оптимизации.

Разработка модуля управления регистратурой на базе технологий React или Angular для фронтенда и Node.js или .NET Core для бэкенда позволит достичь следующих целей:

* Упростить процесс записи на прием и сократить время ожидания для пациентов.
* Повысить эффективность работы врачей за счет оптимизации их расписания.
* Улучшить взаимодействие между различными отделами поликлиники и интеграцию с другими медицинскими системами.
* Создать удобный и интуитивно понятный интерфейс для пациентов и медицинского персонала.

Реализация модуля управления регистратурой позволит значительно улучшить качество обслуживания пациентов и повысить общую эффективность работы поликлиники. В дальнейшем планируется развитие и расширение функционала модуля, что обеспечит его адаптацию к постоянно меняющимся требованиям в сфере здравоохранения.

# Список использованной литературы

1. Шапошников, И. Самоучитель ASP.NET / И. Шапошников. - М.: БХВ-Петербург, 2018. - 368 c.

2. Нейгел, К. C# 2005 для профессионалов / К. Нейгел. - М.: Вильямс, 2020. - 665 c.

3. Хорошилов, А.В. Мировые информационные ресурсы / А.В. Хорошилов, С.Н. Селетков. - М.: СПб: Питер, 2021. - 176 c.

4. Сэнд, Уоррен Hello World! Занимательное программирование / Уоррен Сэнд. - М.: Питер, 2020. - 442 c.

5. Грабер SQL. Справочное руководство / Грабер, Мартин. - М.: Лори; Издание 2-е, 2019. - 354 c.

6. Грофф, Джеймс Р. Энциклопедия SQL / Грофф, Джеймс Р., Вайнберг, Пол Н.. - М.: СПб: Питер, 2021. - 896 c.

7. Макдоналд Oracle PL/SQL практические решения / Макдоналд и др. - М.: СПб: ДиаСофт, 2018. - 560 c.

8. Уорсли, Дж. PostgreSQL. Для профессионалов (+ CD) / Дж. Уорсли, Дж. Дрейк. - М.: СПб: Питер, 2018. - 496 c.

9. SQL Server 2005 шаг за шагом. Практическое руководство. - М.: Эком, 2023. - 463 c.

10. Веймаер, Р. Освой самостоятельно Microsoft SQL Server 2000 за 21 день (+ CD-ROM) / Р. Веймаер, Р. Сотел. - М.: Вильямс, 2022. - 704 c.

11. Аллен Тейлор. "SQL для чайников." М.: Вильямс, 2020. 416 с.

12. Мартишин С., Храпченко М., Симонов В. "Базы данных. Практическое применение СУБД SQL- и NoSQL-типа для применения проектирования." Форум, 2019.

13. Молинаро Э., де Грааф Р. "SQL. Сборник рецептов." 2-е изд. БХВ-Петербург, 2021.

14. Э. Шилдт. "Изучаем C#." 2022. 547 с.

15. Л. Ульман. "MySQL. Руководство по изучению языка." 2023. 1228 с.

# Приложения

# Приложение А – Код создания базы данных

-- Table: ServiceType

CREATE TABLE ServiceType (

    ServiceTypeID INT PRIMARY KEY IDENTITY(1,1),

    Name NVARCHAR(255) NOT NULL

);

-- Table: Specialist

CREATE TABLE Specialist (

    SpecialistID INT PRIMARY KEY IDENTITY(1,1),

    FullName NVARCHAR(255) NOT NULL,

    Position NVARCHAR(255) NOT NULL,

    Experience INT NOT NULL

);

-- Table: Users

CREATE TABLE Users (

    UserID INT PRIMARY KEY IDENTITY(1,1),

    Name NVARCHAR(255) NOT NULL,

    Login NVARCHAR(255) NOT NULL UNIQUE,

    Email NVARCHAR(255) NOT NULL,

    Password NVARCHAR(255) NOT NULL,

    Role NVARCHAR(50) NOT NULL

);

-- Table: Request

CREATE TABLE Request (

    RequestID INT PRIMARY KEY IDENTITY(1,1),

    Date DATETIME NOT NULL,

    ServiceTypeID INT FOREIGN KEY REFERENCES ServiceType(ServiceTypeID),

    SpecialistID INT FOREIGN KEY REFERENCES Specialist(SpecialistID),

    PatientID INT FOREIGN KEY REFERENCES Users(UserID),

    Number NVARCHAR(50) NOT NULL

);

-- Table: Reviews

CREATE TABLE Reviews (

    ReviewID INT PRIMARY KEY IDENTITY(1,1),

    SpecialistID INT FOREIGN KEY REFERENCES Specialist(SpecialistID),

    Title NVARCHAR(255),

    Content NVARCHAR(MAX),

    Date DATETIME NOT NULL

);

# Приложение Б – Код моделей Entity Framework

public class Request

{

    public int RequestID { get; set; }

    public DateTime Date { get; set; }

    public string Number { get; set; }

*// Foreign keys*

    public int ServiceTypeID { get; set; }

    public ServiceType ServiceType { get; set; }

    public int SpecialistID { get; set; }

    public Specialist Specialist { get; set; }

    public int PatientID { get; set; }

    public User Patient { get; set; }

}

public class Review

{

    public int ReviewID { get; set; }

    public string Title { get; set; }

    public string Content { get; set; }

    public DateTime Date { get; set; }

    // Foreign key

    public int SpecialistID { get; set; }

    public Specialist Specialist { get; set; }

}

public class ServiceType

{

    public int ServiceTypeID { get; set; }

    public string Name { get; set; }

    // Navigation property for requests

    public ICollection<Request> Requests { get; set; }

}

public class Specialist

{

    public int SpecialistID { get; set; }

    public string FullName { get; set; }

    public string Position { get; set; }

    public int Experience { get; set; }

*// Navigation properties*

    public ICollection<Request> Requests { get; set; }

    public ICollection<Review> Reviews { get; set; }

}

public class User

{

public int UserID { get; set; }

public string Name { get; set; }

public string Login { get; set; }

public string Email { get; set; }

public string PhoneNumber { get; set; }

public string Password { get; set; }

public string Role { get; set; }

// Navigation property for requests

public ICollection<Request> Requests { get; set; }

}

# Приложение В – Интерфейс приложения

